Agenda for Haskell QuantLib 14/2/14

- Project Constraints
  - **HQL or QuantLib:** Which one to base the code on?
  - **Project goal:** Exact project goal
- Synopsis so far
  - **Project Title:** HQL? is that cheating?
  - **Problem Definition:** Is this still acceptable?
  - **Project Justification:** Is this ok? and do I need more justification?
  - **Time Schedule:** Ugly graphichs asside, is this acceptable?
  - **Bibliography:** What extra information is needed on HQL and QuantLib?

Haskell Quant Library

Synopsis

Student:	Kasper Passov
Supervisors:	Fritz Henglein
	Jost Berthold

Contents

1 Project Title

Haskell Library for Quantitative Analysis

2 Problem Definition

I want to design and develop a Haskell library for quantitative finance, taking the open source quantlib as a starting point. The language Haskell is used because of its purity and advanced type class system which allows it to model relevant entities.
The result of my project should be a software architecture that supports different kinds of financial instruments and valuation methods.

3 Project Constraints

TODO: Find the freaking constraints

4 Project Justification

TODO: further describition needed

4.1 Reasoning for module (TAKEN FROM Quantlib[2])

Such a library could greatly improve the workout of quants (do i need to include this?).
Few good resources for quantative libraries appart from Quadlib

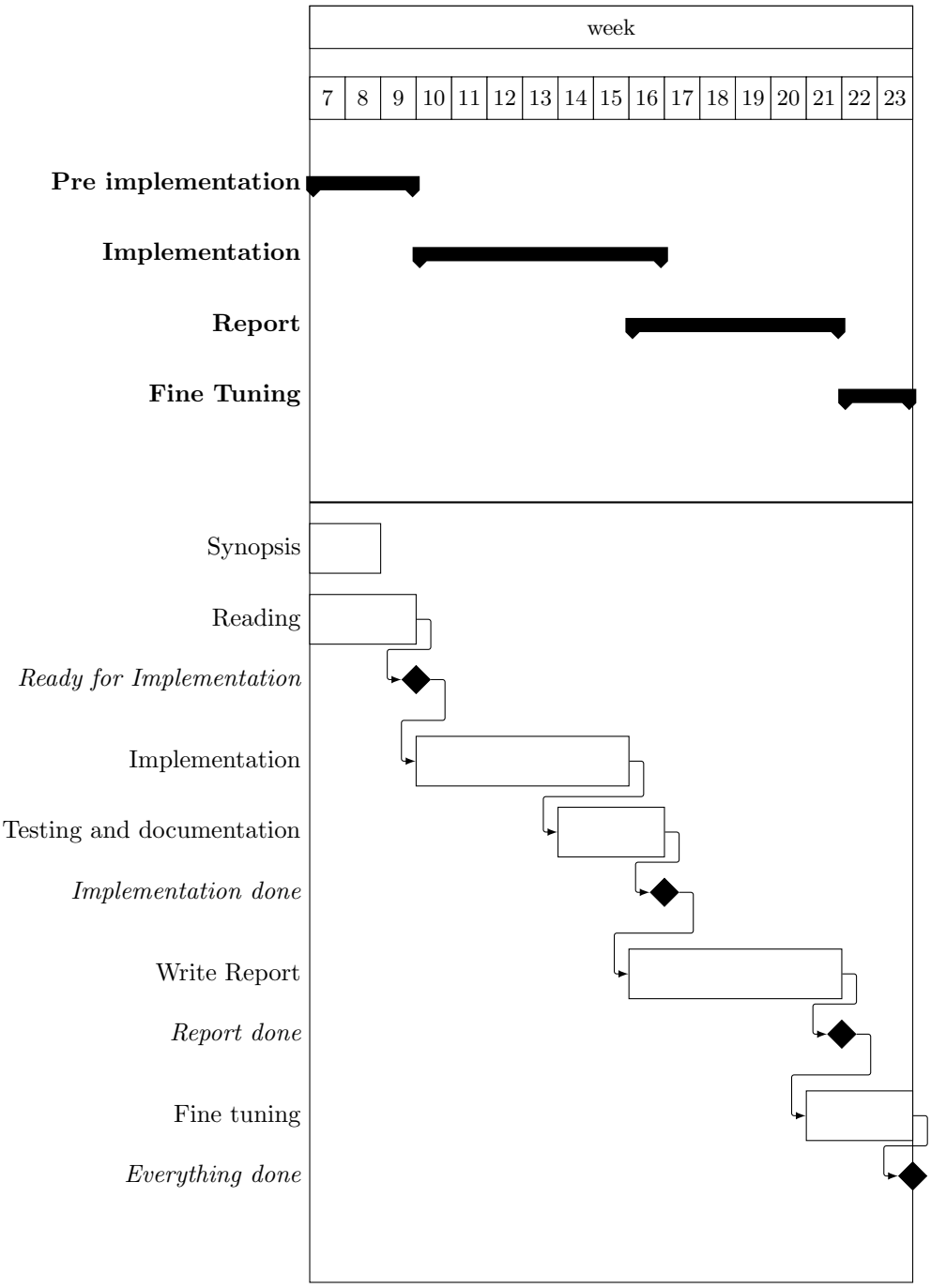
4.2 Reasoning for Haskell (TAKEN FROM HQL[3])

modularity
laziness
Haskell is a pure language like the math behind the methods making implementation easier.
Functional programming, making it easier to reuse functions for multiple tasks
Statically type, errors on compile and overload types (not 100% sure how this works yet, is this the type families?)

5 Schedule

5.1 Timeline

TODO: more precision based on constraints, also description of legends


week
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Pre implementation
Implementation
Report
Fine Tuning
Synopsis
Reading
Ready for Implementation
Implementation
Testing and documentation
Implementation done
Write Report
Report done
Fine tuning
Everything done

5.2 Work Activities and tasks

Activities and tasks as described in OOSE[1]
TODO: more serious description of activities and tasks.

5.2.1 Activities

Pre implementaion Before i implement the code i need a lot of knowledge.

Implementaion The time used to implement the project describer in constraints

Report The time to write everything i think i know about my code. (maybe let this overlap more with the implementation?)

Fine Tuning Wastetime so the schedule has room for sliding

5.2.2 Tasks

Synopsis Write a synopsis

Reading Read stuff when needed. Proberly a lot longer

Implementation Implement the code. Needs to be split in smaller pieces when i have some constraints

Testing and documentation Write and run tests on the implementation, and write documentation

Write report Write the report, also needs to be split further

Fine tuning Time to correct any and all spelling errors, and maybe any coding and logical errors if i did not get that right the first time

References

[1] Bernd Bruegge, Allen H.Dutoit *Object-Oriented Software Engineering Using UML, Patterns and Java*, Person New International Edition, Third Edition, 2014.

[2] Luigi Ballabio *Implementing Quantlib*, Draft, 2013.

[3] Andreas Bock, johan Astborg, Jost Berthold, Sinan Gabel *HIPERFIT Quant Library*, 2014.